Programación Python para Machine Learning

Lección 9: Árboles de Decisión.

eip ESCUELA INTERNACIONAL POSGRADOS

ÍNDICE

Lección 9: Árboles de Decisión.

1. Introducción

Bajo los intuitivos principios en los que se basan los Árboles de Decisión, se encuentra uno de los modelos de supervisados de Machine Learning más simples y poderosos.

Debido a que son modelos no lineales y muy interpretables, es uno de los modelos con mayor aceptación dentro del área para completar tareas predictivas tanto de clasificación como de regresión.

Esta lección está destinada a abordar conceptos como qué son los Árboles de decisión, sus principios teóricos, y cómo implementarlos en Python.

Objetivos

- Conocer los principios en los que se basan los Árboles de decisión.
- Reconocer los componentes de un Árbol de decisión y entender cómo se construye.
- Dominar las técnicas de implementación de los Árboles de decisión en Python.
- Identificar los aspectos a tener en cuenta para mejorar el rendimiento de un Árbol de decisión.

2. Principios teóricos de los Árboles de Decisión

Los Árboles de Decisión son un modelo supervisado no lineal de Machine Learning que pueden ser utilizados para problemas de clasificación y regresión.

La idea que hay detrás de los árboles de decisión es sencilla e intuitiva. De modo iterativo se deben ir generando particiones binarias de los datos de modo que cada nueva partición genere un subgrupo de datos lo más homogéneo posible.

El árbol de decisión es una estructura que está formada por ramas y nodos de distintos tipos:

- Los **nodos** internos representan cada una de las características a considerar para tomar una decisión y generar la partición.
- Las **ramas** representan la decisión en función de la condición del nodo del que parten.
- Los **nodos hoja** representan el resultado final de la decisión, es decir, la predicción.

El punto de partida del árbol contiene la primera condición para realizar la partición. Es el conocido como **nodo raíz**. La **profundidad** del árbol es la trayectoria entre la raíz y el nodo hoja más lejano.

Para cuantificar la homogeneidad en se pueden utilizar varias métricas dependiendo de si se trata de un árbol para abordar un problema de clasificación o para un problema de regresión:

- Clasificación:
  - Índice de Gini
  - Entropía (Ganancia de información)
- Regresión:
  - Error cuadrático medio (MSE)
  - Error absoluto

Ante la posibilidad de diversas posibles particiones, el algoritmo debe decidir cuál de ellas es la más adecuada. Para ello define una **función de costo** que asigna al nodo padre un promedio ponderado de la métrica de homogeneidad de sus dos nodos hijo.

El proceso de entrenamiento de un árbol de decisión sigue el siguiente esquema:

1. Para decidir el nodo raíz del árbol, es decir, la primera partición, se toman todas las características del conjunto de entrenamiento y, para cada una de ellas, se definen todos los posibles umbrales. Se considera umbral cada punto intermedio entre dos valores consecutivos de cada característica.
2. Para cada uno de los umbrales, se calcula la partición que surge y que conforma los dos hijos. Se calcula la métrica de homogeneidad o pureza para cada hijo y con dichos valores, la función de costo del nodo padre.
3. Aquel umbral con menor función de costo es el elegido, ya que la partición obtenida es la más homogénea de todas.
4. El proceso se repite de modo iterativo hasta conseguir todos los nodos hoja.

Este tipo de algoritmo de entrenamiento es un método voraz en tanto que va tomando la decisión más óptima en cada caso, sin poder considerar si de manera global el rendimiento del árbol es óptimo igualmente.

Los árboles que terminan con todos los nodos hoja “puros” tienen mucha probabilidad de padecer de *overfitting*. Ante eso, existen dos posibles estrategias:

- Estrategia pre-pruning: Consiste en restringir el crecimiento del árbol durante el entrenamiento, por ejemplo, limitando la profundidad del árbol.
- Estrategia post-pruning: Consiste en quitar nodos una vez entrenado.

Una de las grandes ventajas de los Árboles de Decisión es la facilidad con la que pueden ser interpretado el modelo tras el entrenamiento. Gráficamente, resulta relativamente fácil identificar el conjunto de reglas que permiten que un dato sea predicho en una categoría u otra.

Esta situación es especialmente fácil cuando el número de características es moderado. Además, también puede apreciarse qué características son más relevantes para la predicción.

3. Implementación en Python de un Árbol de Decisión

Los Árboles de Decisión presentan una versatilidad dentro del Machine Learning que los hacen ser un modelo apropiado para resolver tanto problemas de clasificación como problemas de regresión.

Se puede llevar a cabo la construcción de un modelo de Árbol de decisión en Python utilizando el módulo `tree` de scikit-learn.

En todo caso, hay que recordar que cuando se trata con un modelo de Árboles de Decisión en scikit-learn, todas las variables deben ser de numéricas. Por tanto, toda variable categórica en el conjunto de datos debe ser convertidas en numéricas del modo más adecuado y representativo posible.

3.1. Árboles de Decisión para clasificación

Scikit-Learn cuenta con la clase `DecisionTreeClassifier` dentro de su módulo `tree`, que permite crear un árbol de decisión para proceder con problemas de clasificación.

Se hará uso del conjunto de datos *magic* del repositorio *UCI Machine Learning*.

El parámetro de la clase `DecisionTreeClassifier` fundamental que configura una estrategia prepoda es `max_depth`. Para configurar una estrategia postpoda, el parámetro adecuado es `ccp_alpha`.

En todo caso, hay que recordar que cuando se trata con un modelo en scikit-learn, todas las variables deben ser de numéricas.

Por tanto, toda variable categórica en el conjunto de datos debe ser convertidas en numéricas del modo más adecuado y representativo posible.

```
from pandas import read_csv
from sklearn.tree import DecisionTreeClassifier, plot_tree
import time, random
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import confusion_matrix, accuracy_score,
balanced_accuracy_score

seed=random.seed(time.time())
filename = '../datasets/magic04.data'

col_names=['fLength','fWidth','fSize','fConc','fConc1','fAsym','fM3Lon
g','fM3Trans','fAlpha','fDist','class']
data = read_csv(filename, names=col_names)

X = data[data.columns[:-1]]
Y = data['class']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =
0.3, random_state=seed)


model = DecisionTreeClassifier(max_depth=5).fit(X_train, y_train)
y_pred = model.predict(X_test)

bacc =  balanced_accuracy_score(y_pred, y_test)
acc =  accuracy_score(y_pred, y_test)
cm =  confusion_matrix(y_pred, y_test)
plot_tree(model)

print(bacc, acc)
print(cm)
```

3.2. Árboles de Decisión para regresión

Scikit-Learn cuenta con la clase `DecisionTreeRegressor` dentro de su módulo `tree`, que permite crear un árbol de decisión para proceder con problemas de regresión. Se hará uso del conjunto de datos *housing* del repositorio *UCI Machine Learning.*

El parámetro de la clase `DecisionTreeClassifier` fundamental que configura una estrategia prepoda es `max_depth`. Para configurar una estrategia postpoda, el parámetro adecuado es `ccp_alpha`.

En todo caso, hay que recordar que cuando se trata con un modelo en scikit-learn, todas las variables deben ser de numéricas. Por tanto, toda variable categórica en el conjunto de datos debe ser convertidas en numéricas del modo más adecuado y representativo posible.

```
from pandas import read_csv
from sklearn.tree import DecisionTreeRegressor, plot_tree
import time, random, numpy
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import mean_squared_error

seed=random.seed(time.time())
filename = '../datasets/housing.csv'

col_names=['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',
'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
data = read_csv(filename, names=col_names,sep=" ")

X = data[data.columns[:-1]]
Y = data['MEDV']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =
0.3, random_state=seed)

model = DecisionTreeRegressor(max_depth=20,
ccp_alpha=0.5).fit(X_train, y_train)
y_pred = model.predict(X_test)
y_pred_train = model.predict(X_train)

rmse= numpy.sqrt(mean_squared_error(y_train, y_pred_train))
print(rmse)
rmse= numpy.sqrt(mean_squared_error(y_test, y_pred))

print(rmse)
plot_tree(model)
```

4. CONSEJOS PRÁCTICOS SOBRE LOS ÁRBOLES DE DECISIÓN

4.1. Clasificación multiclase

Por su diseño, los Árboles de Decisión tienen una naturaleza que les permite poder resolver problemas de clasificación binarios y problemas de clasificación multiclase.

Por tanto, la clase `DecisionTreeClassifier` de scikit-learn implementa la clasificación multiclase de manera que queda totalmente transparente al programador esta característica. Se hará uso del conjunto de datos *iris* del repositorio *UCI Machine Learning.*

El parámetro de la clase `DecisionTreeClassifier` fundamental que configura una estrategia prepoda es `max_depth`. Para configurar una estrategia postpoda, el parámetro adecuado es `ccp_alpha`.

```
from pandas import read_csv
from sklearn.tree import DecisionTreeClassfier, plot_tree
import time, random, numpy
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

filename = '../datasets/iris.data'
col_names=[  'sepal length', 'sepal width','petal length','petal
width','class']
data = read_csv(filename, names=col_names)

X = data[data.columns[:-1]]
Y = data['class']


X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =
0.5, random_state=seed)

model = DecisionTreeClassifier(max_depth=5).fit(X_train, y_train)
y_pred = model.predict(X_test)

bacc =  balanced_accuracy_score(y_pred, y_test)
acc =  accuracy_score(y_pred, y_test)
cm =  confusion_matrix(y_pred, y_test)

plot_tree(model)
print(bacc, acc)
print(cm)
```

4.2. Consideraciones a tener en cuenta

Los Árboles de Decisión, debido a su naturaleza, tienen una serie de características que los hacen ser un modelo a tener en cuenta a la hora de realizar un proyecto de Machine Learning:

- Requieren de una escasa preparación de los datos. En comparación con otras técnicas que suelen requerir de la normalización de los datos, transformación de las variables categóricas o tratamiento de los valores faltantes.
- Se debe considerar que el coste computacional del algoritmo es logarítmico respecto al número de instancias utilizadas para entrenar el árbol.
- Utiliza un modelo de caja blanca. Si una situación determinada es observable en un modelo, la explicación de la condición se explica fácilmente mediante la lógica booleana. Por el contrario, los modelos de caja negra (SVM, NN...), este tipo de sucesos son más difíciles de interpretar.
- Este tipo de modelos no asumen ninguna característica en la distribución de los datos.

Sin embargo, hay que tener presente una serie de consideraciones si se desea un mejor rendimiento:

- Los modelos de árboles de decisión pueden crear árboles demasiado complejos que no generalizan bien los datos. Para evitar este problema es necesario aplicar los mencionados mecanismos de poda.
- Los árboles de decisión pueden ser inestables. Ante la más mínima variación en los datos de entrada, se genera un árbol completamente diferente. Este problema se puede mitigar utilizando árboles de decisión dentro de un ensemble.

- Entrenar un árbol decisión con algoritmos voraces, en el que se toman decisiones localmente óptimas en cada nodo, no garantiza que devuelvan el árbol de decisión globalmente óptimo. Igualmente, esta problemática puede mitigarse entrenando múltiples árboles en un ensemble.
- Los modelos de árboles de decisión crean árboles sesgados si algunas clases dominan. Por lo tanto, se recomienda equilibrar el conjunto de datos antes de ajustarlo con el árbol de decisión.

5. PUNTOS CLAVE

En esta lección se ha aprendido:

- Conocer los conceptos generales de los Árboles de Decisión.
- Detallar los pasos que se siguen para construir los Árboles de Decisión
- Implementar en scikit-learn un modelo de Árbol de Decisión tanto para problemas de clasificación binaria como multiclase.
- Implementar en scikit-learn un modelo de Árbol de Decisión tanto para problemas de regresión.
- Profundizar en aquellos aspectos de un Árbol de Decisión que pueden ser determinantes en el rendimiento de este tipo de modelos.